

NASA TM-83173

NASA Technical Memorandum 83173

NASA-TM-83173 19820002879

PERSONAL COPY

NASA LARC RATFOR DOCUMENTATION

VERSION 1.0

H. J. Dunn

September 1981



National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665



NF00233

NASA LaRC RATFOR DOCUMENTATION VERSION 1.0

H. J. Dunn

INTRODUCTION

The purpose of this paper is to describe the use of a preprocessor at the LaRC computer center that converts RATFOR source into FORTRAN source code that complies with the ANSI 1966 FORTRAN standard. A FORTRAN source of an existing RATFOR preprocessor was converted to run on LaRC computers. In order for this to be done, some minor changes were made to the syntax of the language. This paper describes the RATFOR preprocessor that is implemented at LaRC. The philosophy on which RATFOR is based and more details on the language can be found in reference 1. The primary purpose of RATFOR is to make FORTRAN a better programming language, for both writing and structuring programs. This is done by providing the control structures that are unavailable in bare FORTRAN, and by improving the "cosmetics" of the language. By writing programs in RATFOR, they will be easier to understand and easy to change if the need arises.

The control flow structures of RATFOR are DO, FOR, IF, REPEAT, WHILE, BREAK, NEXT and statement groupings with brackets. These structures permit programming without the use of GOTO statements and result in code that is easier to read. The cosmetic aspect of RATFOR has been designed to make it concise and reasonably pleasing to the eye. It is free-form in that statements may appear anywhere on an input line. The end of a line generally marks the end of a statement, but lines that are obviously not finished, such as lines ending with a comma, automatically continue onto the next line. Multiple statements may appear on one line if separated by semicolons. The comment convention, a sharp # anywhere in a line signals the beginning of a comment and helps to encourage unobtrusive marginal remarks. Quoted strings are converted into H's. Notations like ">" convey the meaning of "greater than" more rapidly than equivalent forms like .GT. Simple string replacement macro's (DEFINE's) and conditional processing (IFDEF / IFNOTDEF / ENDIFDEF) are incorporated so as to increase the portability of programs written in RATFOR.

With these two aspects, flow control and cosmetics, RATFOR can generate a well-structured program with source code that is easy to follow. This will make the program easier to develop and in turn will result in more reliable results.

THE RATFOR LANGUAGE

In the following description of the RATFOR language the term "statement" can either be a RATFOR single or compound statement. A description of the compound statement is given in the RATFOR language features section. Since an objective of this report is to provide a reference document for the language, and not a tutorial, the language elements are listed in alphabetical order.

1. BREAK

The break statement causes an immediate exit from an enclosing DO, FOR, REPEAT or WHILE loop and continues at the first statement following the loop. Only one loop can be terminated by a BREAK, even if the BREAK is contained inside several nested loops. Examples of the use of the BREAK statement can be found in the examples for each of the looping statements.

2. DEFINE (symbol=replacement string)

Each occurrence of a defined symbol in the program or INCLUDE file (see part 6 of this section) is replaced by the "replacement string," which is then processed as input to the processor. The definition of a symbol constant can be another defined symbol. Once a symbol is defined, it cannot be redefined. Defined symbols must be unique alphanumeric character strings beginning with a letter. There are no special characters or blanks allowed. Replacement strings can be any character string less than 70 characters long but must not contain dollar signs and must fit on one source line.

The replacement string, either in whole or in part, can contain a simple integer mathematical relationship between one or more defined symbols and/or integers enclosed in less-than and greater-than symbols (<,>). Addition, subtraction, multiplication and division are allowed. Evaluation of the expression is strictly left to right. There is no hierarchy of the operators. If the expression contains previously defined constants, they are replaced by their replacement string prior to mathematical evaluation.

EXAMPLE:

DEFINE(LEXBREAK=-110)	}	in RATFOR becomes
DEFINE(LEXDIGITS=<LEXBREAK-1>)		
B=LEXBREAK		
A=LEXDIGITS		
B=-110	}	in the FORTRAN code.
A=-111		

The passing of a single argument to be included in the replacement string is distinguished from a simple DEFINE by the presence of at least one dollar sign in the definition. Any occurrence of a dollar sign in the replacement text will be replaced by the argument of the defined symbol when it is actually called.

EXAMPLE:

DEFINE(INCREMENT,\$=\$+1)	}	in RATFOR becomes
INCREMENT(I)		
I = I + 1	}	in the FORTRAN code.

3. DO index=limits statement

The DO statement sets up a standard FORTRAN DO loop. The "limits" must be a legal FORTRAN DO specification since it is copied into the FORTRAN code directly. RATFOR supplies the appropriate statement number.

EXAMPLE:

DO I=1,10 [IF(I==9) BREAK IF(I==2) NEXT K=I-2]	}	in RATFOR becomes
DO 20000 I=1,10 IF (.NOT.(I.EQ.9)) GOTO 20002 GOTO 20001 20002 CONTINUE IF (.NOT.(I.EQ.2)) GOTO 20004 GOTO 20000 20004 CONTINUE K=I-2 20000 CONTINUE 20001 CONTINUE	}	in the FORTRAN code.

4. FOR (initialize; condition; reinitialize) statement

The "initialize" statement is executed, then the "statement" and "reinitialize" are executed as long as "condition" is true. The "condition," "initialize" and "reinitialize" parts are single FORTRAN statements. The "condition" is tested before each iteration. Any of the three parts may be omitted, although the semicolons must remain. A null "condition" is treated as always true, so that an infinite loop results when the "condition" is omitted.

EXAMPLE:

FOR (X=0.; X<=1.; X=X+.05) [Y=EXP(X) IF(Y==0) NEXT IF(Y>=12.) BREAK Z=EXP(Y)]	}	in RATFOR becomes

```

      X=0.
20006 IF (.NOT.(X.LE.1.)) GOTO 20008
      Y=EXP(X)
      IF (.NOT.(Y.EQ.0)) GOTO 20009
      GOTO 20007
20009 CONTINUE
      IF (.NOT.(Y.GE.12.)) GOTO 20011
      GOTO 20008
20011 CONTINUE
      Z=EXP(Y)
20007 X=X+.05
      GOTO 20006
20008 CONTINUE

```

} in the FORTRAN code.

5. IF (condition)

statement-1

ELSE

statement-2

The ELSE and statement-2 are optional. If the "condition" is true, statement-1 is executed; if it is false and there is an ELSE clause, statement-2 is executed. In the absence of brackets, each ELSE goes with the previous un-ELSEd IF.

EXAMPLE:

```

IF(I==J) A=1.
ELSE A=2.

```

} in RATFOR becomes

```

      IF (.NOT.(I.EQ.J)) GOTO 20013
      A=1.
      GOTO 20014
20013 CONTINUE
      A=2.
20014 CONTINUE

```

} in the FORTRAN code.

6. INCLUDE/NL filename

When the RATFOR program encounters an INCLUDE statement, the contents of the local file with the name "filename" are read in as source and processed. When the end of the file is reached, the input of the preprocessor reverts to the next line of the original file. The include file may be nested three deep. The /NL is an optional "no-list" switch used to suppress the listing of the file in the output listing. Any include statements within the file with the no-list switch specified will not be listed, regardless of the switch settings on their INCLUDE lines.

7. NEXT

The rest of the containing loop is skipped and program continues with the next iteration of the loop. For the DO, REPEAT...UNTIL and WHILE statements the control is to the "condition" test; for the FOR statement, the control is to the "reinitialize" statement; and for an infinite REPEAT, the control is to the top of the loop. Examples of the NEXT can be found in the examples for each looping element.

8. null statement

; (used by itself)

The semicolon may be used anywhere that another RATFOR statement may be used.

9. REPEAT statement

UNTIL (condition)

The "statement" is executed until the "condition" is true. The "condition" is a single FORTRAN statement that is tested after each iteration. The UNTIL statement is optional and if omitted the result is an infinite loop.

EXAMPLE:

```
REPEAT [  
    A=A+1.  
    IF(A==7.) NEXT  
    Y=F(A)  
    IF(Y==0) BREAK  
    ]  
UNTIL (A==0. ++ Y==100.)
```

} in RATFOR becomes

```
20015 CONTINUE  
    A=A+1.  
    IF (.NOT.(A.EQ.7.)) GOTO 20018  
    GOTO 20016  
20018 CONTINUE  
    Y=F(A)  
    IF (.NOT.(Y.EQ.0)) GOTO 20020  
    GOTO 20017  
20020 CONTINUE  
20016 IF (.NOT.(A.EQ.0..OR.Y.EQ.100.)) GOTO 20015  
20017 CONTINUE
```

} in the FORTRAN code.

10. WHILE (condition)

statement

The "statement" is executed as long as the "condition" is true. The "condition" is tested before each iteration.

EXAMPLE:

WHILE (B<=3) [X=XYZ(B) IF(X==2) BREAK Y=X+3 IF(Y==2.5) NEXT Z=Y-78.]	}	in RATFOR becomes
20022 IF (.NOT.(B.LE.3)) GOTO 20023 X=XYZ(B) IF (.NOT.(X.EQ.2)) GOTO 20024 GOTO 20023 20024 CONTINUE Y=X+3 IF (.NOT.(Y.EQ.2.5)) GOTO 20026 GOTO 20022 20026 CONTINUE Z=Y-78. GOTO 20022 20023 CONTINUE	}	in the FORTRAN code.

RATFOR LANGUAGE FEATURES

1. COMMENTS

A sharp sign # used anywhere on a line causes the rest of the line to be treated as a comment. The sharp sign may occur in the first column, if desired, replacing the FORTRAN "C" in column one. In this case, the entire line is converted to uppercase and copied into the FORTRAN code as a comment (unless the /CO switch is in effect; see the Command Line Options section).

2. COMPOUND STATEMENT

Brackets [], can be used to enclose single or multiple RATFOR and/or FORTRAN statements so that the enclosed block of statements may be used anywhere that a single RATFOR statement may be used.

3. RATIONAL AND LOGICAL OPERATORS

Since symbols are clearer than the .EQ., .GT., etc. used by FORTRAN, RATFOR allows the use of conventional mathematical symbols. These symbols are converted into the equivalent FORTRAN according to the following:

> for .GT.
== for .EQ.
>= for .GE.
\= for .NE.
< for .LT.
<= for .LE.
\ for .NOT.
++ for .OR.
& for .AND.

4. CONTINUATION LINES

RATFOR source code lines are automatically continued if:

1. The statement is obviously incomplete at the end of the line, as in the middle of the conditional part of a FOR or IF statement.
2. The line ends with a comma.
3. The line ends with an underline character "_" (the underline character is not passed to the FORTRAN output).

5. QUOTED STRINGS

Quoted strings are converted into the equivalent Hollerith string.

6. IFDEF / IFNOTDEF / ENDIFDEF (Conditional Processing)

Sections of RATFOR code (one or more lines) can be selectively processed into FORTRAN or ignored, depending upon the current define symbol status of a specific constant. When "IFDEF(symbol)" is encountered in the RATFOR source code, a check is made to see if "symbol" has previously appeared in a DEFINE statement; if it has, the source code up to the balancing ENDIFDEF statement is processed; if not, the source code is skipped until the balancing ENDIFDEF is found. The "IFNOTDEF(symbol)" is similar, except that the RATFOR source code up to the balancing ENDIFDEF is processed if "symbol" has not been previously defined.

The symbolic constant can be given a null definition, if it is being defined only for use with the INDEF/IFNOTDEF statements (e.g. DEFINE (foo=) is sufficient). IFDEFs (and IFNOTDEFs) can be nested; if an outer conditional is unsatisfied, all inner conditionals are skipped, just like all other code within the unsatisfied conditional.

Undefined conditional code (that not processed into FORTRAN) is normally printed in the RATFOR source listing, but will have no source code line numbers on the left-hand side of the page. The /IF command line option can be used to suppress the listing of the undefined conditional code.

7. LITERAL LINES (%)

If a percent sign (%) occurs in column one of a RATFOR source code line, the entire line except for the percent sign will be passed to the FORTRAN code without any modification whatsoever.

8. DEBUG LINES (?)

If a question mark (?) occurs in column one of a RATFOR source code line, it is considered to be a debug line and will be processed into FORTRAN (minus the question mark) only if the /DE (DEBUG) switch was specified in the command line (see next section). Multiple levels of debug statement can be specified by a digit (1-9) in the second column (after the "?"). Debug lines whose level is equal to or greater than the level specified in the /DE:n switch are processed, but lines with a lower level are not processed into FORTRAN. Lines with no level specified (blank in column two) are always processed if the /DE switch is specified. A /DE switch with no value causes all debug lines to be processed. For example, the line:

```
?3 PRINT X
```

would print the value of the variable X only if the command line contained a /DE or /DE:n, with n less than or equal to 3.

9. STRINGS

Since character processing frequently requires the use of strings, the preprocessor adds the STRING data type to FORTRAN. In FORTRAN, a STRING becomes an integer array with one character per element, plus one element for the terminator (end-of-string character). The number that is assigned to each work is the value of the ASCII character code.

EXAMPLE:

```
STRING FOO "BLATZ"
```

becomes INTEGER FOO(6)

```
DATA FOO/66,76,65,84,90,eos/
```

Note that the STRING function requires that the symbolic constant "eos" be defined when the STRING keyword is first encountered; otherwise "eos" will be passed to the FORTRAN code as is and upset the compiler.

Since ANSI standard FORTRAN requires that all DATA statements must be grouped together and placed in the FORTRAN code after all other specification statements, but before any executable statements, STRING statements must be grouped together and appear in the RATFOR source code after all other specification statements but before any DATA statements. The preprocessor holds all the DATA statement parts until the "integer" statement parts for all STRINGS have been transmitted to the outer file and then outputs the DATA statements as a group. There is a limit of 12 string specifications statements with a total of 150 characters in any one program module.

COMMAND LINE OPTIONS

The command line switches are available to control the actions of the preprocessor. The command switches must be contained in a comment line that is first recorded in the file. This comment is printed as the second line in the heading of the output listing. Where appropriate, a switch can be negated by /NOSw or /-sw. The following are the command line switches:

/CO compress	Causes the FORTRAN code that is generated by the preprocessor to be compressed for faster I/O by eliminating all comments and unnecessary blanks in the generated FORTRAN code. Default: /NOCO.
/DE:n debug	Causes all lines beginning with a question mark in column one to be processed into FORTRAN code; by default such lines are ignored. If n is specified, only debug lines with an equal or higher value in column two will be processed.
/FO fortran	Causes the generated FORTRAN code to be included at the end of the listing. Default: /NOFO.
/FT ftn	Generate FORTRAN source code. Default: /FT.
/IF ifdef	Causes RATFOR source code within unsatisfied conditionals (IFDEFs that are not defined or IFNOTDEFs that are defined) not to be printed in the listing file, except for the IFDEF or IFNOTDEF statement. Default: /NOIF.
/LC lower case	Cause the generated FORTRAN code to be in lower case characters. Default: /NOLC.
/LI list	Generate the RATFOR listing. Default: /LI.
/SY symbols list	List the defined symbols table. Default: /NOSY.

PREPROCESSOR USE UNDER NOS

The RATFOR processor is run by using the following NOS commands:

```
GET,RATFOR/UN=236939N.
```

```
RATFOR(INPUT,OUTPUT,COMP)
```

The three files that are used by RATFOR have the default names of INPUT, OUTPUT and COMP. The RATFOR source code is read into the processor on the INPUT file, and must not have a record length greater than 181 characters. If a different command line is to be used for more than one program module, these modules must be separated by an EOF mark. This can be done by using the COPYBF command to copy each file into a temporary file and passing this file to the RATFOR processor. The RATFOR listing is written to the OUTPUT file and the FORTRAN source code is written to the COMP file.

After a program has been processed by RATFOR, the program control registers R1 and/or EF may be examined to determine if there were any detected errors in the RATFOR source code. If an error or errors have occurred in the source code, but the preprocessor was able to process the entire file, R1 will be 1 and EF will be 0. If the preprocessor has had to abort, R1 will be 1 and EF will be 4. The distinction is made because in the first case the user may want to correct the FORTRAN source and continue. Being able to do this in the second case is very optimistic. If no errors were detected, R1 and EF are 0.

PROGRAM EXAMPLE

The following program gives an example of RATFOR. It is not exhaustive but should help in the understanding and use of RATFOR. The program selected for the example will copy the input file to the output until an end-of-file condition is raised. Since the internal ASCII character set is used, lower case and terminal control characters may be transmitted to and from a TELEX terminal with relative ease.

```

1      # PROGRAM FOR RATFOR EXAMPLE /FO/SY
2      INCLUDE SETUP
3      *   DEFINE(MAXLINE=91) # MAX WIDTH OF OUTPUT LINE
4      *   DEFINE(DECREMENT, $=$-1)
5      *   DEFINE(INCREMENT, $=$+1)
6      *   DEFINE(MAX=80) # MAX STRING LENGTH
7      *   DEFINE(MAXDISPLAY=<MAX+1>) # MAX DISPLAY LENGTH
8      *   DEFINE(BUFFERLEN=<3*MAXDISPLAY+3>/<MAXDISPLAY+1>) # INPUT/OUTPUT BUFFER SIZE
9      *   DEFINE(BAD=-1)
10     *   DEFINE(YES=1)
11     *   DEFINE(EOF=-3)
12     *   DEFINE(EOS=0)
13     *   DEFINE(DUMMYSIZE=1)
14     *   DEFINE(TAPEND=TAPE$)
15     *   DEFINE(STDIN=5) # STANDARD INPUT UNIT NUMBER
16     *   DEFINE(STDOUT=6) # STANDARD OUTPUT UNIT NUMBER
17     *   DEFINE(HUGE=32767)
18     INCLUDE/NL CHAR
19     PROGRAM EXRAT(INPUT=BUFFERLEN, # INPUT FILE
20                OUTPUT=BUFFERLEN, # OUTPUT FILE
21                TAPEND( STDIN ) = INPUT, # INPUT TAPE NUMBER
22                TAPEND( STDOUT ) = OUTPUT ) # OUTPUT TAPE NUMBER
23
24     INTEGER IQ, BUFFER(MAXDISPLAY)
25     INTEGER STRPUT, STRGET
26     STRING START "START OF INPUT"
27     IQ=STRPUT(STDOUT, START)
28     REPEAT [
29       IQ=STRGET(STDIN, BUFFER, MAX) # GET INPUT STRING
30       IF(IQ == EOF) BREAK # STOP IF DONE
31       IQ=STRPUT(STDOUT, BUFFER) # WRITE STRING TO OUTPUT
32     ]
33     WRITE(STDOUT, 1)
34     1 FORMAT(10X, "END OF INPUT")
35
36     STOP
37     END

```

```

#
# SLEN - COMPUTE LENGTH OF STRING
#
42     INTEGER FUNCTION SLEN(STR)
43     INTEGER STR(DUMMYSIZE)
44     DO SLEN=1, HUGE
45       IF(STR(SLEN) == EOS) BREAK
46     DECREMENT(SLEN) # WENT 1 TOO FAR
47     RETURN
48     END

```

PROGRAM FOR RATFOR EXAMPLE /FD/SY

```

#
# STRPUT - WRITE A STRING TO A SPECIFIED LUN
#HJ DUNN APRIL 14, 1981
#
49      INTEGER FUNCTION STRPUT(LUN,STR)
50      INTEGER LUN,I,N,SLEN,MINO,CEOF
51      INTEGER STR(DUMMSIZE), BUFF(MAXDISPLAY)
#
52      I=MINO(MAXLINE,SLEN(STR))
53      IF(STR(1) == FORMFEED) [
54          DECREMENT(I)
55          IF(I == 0) [
56              WRITE(LUN,2) # NEWPAGE ONLY
57              RETURN ]
58          CALL OUTMAP(STR(2),I,BUFF,N)
59          WRITE(LUN,2) (BUFF(I),I=1,N) # NEW PAGE WITH HEADER
60          2 FORMAT(141, MAXDISPLAY R1 )
61      ]
62      ELSE [
63          IF( I == 0 ) [
64              WRITE(LUN,1) # BLANK LINE
65              RETURN ]
66          ELSE [
67              CALL OUTMAP(STR,1,BUFF,N)
68              WRITE(LUN,1) (BUFF(I),I=1,N) # PRINT SINGLE LINE
69              1 FORMAT(1X, MAXDISPLAY R1)
70          ]
71      ]
72      IF( CEOF(LUN) \= 0 ++ IOCHEC(LUN) \= 0)
73          STRPUT=BAD
74      ELSE
75          STRPUT=YES
76      RETURN
77      END

```

PROGRAM FOR RATFOR EXAMPLE /FD/SY

```

#
# STRGET - READ A STRING FROM A SPECIFIED LUN
#
78      INTEGER FUNCTION STRGET(LUN, STR, MAXS)
79      INTEGER LUN,STR(MAXS),BUFF(MAXDISPLAY)
80      READ(LUN,1) BUFF
81      1 FORMAT(MAXDISPLAY R1)
82      IF(CEOF(LUN) \= 0 ) [
83          STRGET=EOF
84          STR(1)=EOS
85          RETURN
86      ]
87      IF(IOCHEC(LUN) \= 0 ) [
88          STRGET=BAD
89          STR(1)=EOS
90          RETURN
91      ]
92      CALL INMAP(BUFF,MAXDISPLAY,STR,MAXS) # CONVERT DISPLAY TO ASCII
93      RETURN
94      END

```

```

#
# CEUF - HIDDEN EOF FUNCTION
#HJ DUNN, APRIL 14, 1981
#
95     INTEGER FUNCTION CEUF(LUN)
96     ?     INTEGER EOF,LUN
97     ?     CEUF=EOF(LUN)
98     RETURN
99     END

```

```

#
# INMAP - CONVERT DISPLAY CODE TO INTERNAL ASCII
#HJ DUNN, APRIL 10, 1981
#
100    SUBROUTINE INMAP(STR1,MAX1,STR0,MAX0)
101    INTEGER STR1(DUMMYSIZE),MAX1, # INPUT STRING (LEN=MAX1)
102    STR0(DUMMYSIZE),MAX0, # OUTPUT STRING (LEN=MAX0)
103    DSPLY(2,64),DSPLY1(8) # DISPLAY CODE INPUT CHAR
104    INTEGER J
105    DATA DSPLY1 /0,64,94,37,58,3#0/
106    DATA DSPLY1 / 53, 96, 65, 97, 66, 98, 67, 99, 68, 100,
107    69, 101, 70, 102, 71, 103, 72, 104, 73, 105,
108    74, 106, 75, 107, 76, 108, 77, 109, 78, 110,
109    79, 111, 80, 112, 81, 113, 82, 114, 83, 115,
110    84, 116, 85, 117, 86, 118, 87, 119, 88, 120,
111    89, 121, 90, 122, 48, 123, 49, 124, 50, 125,
112    51, 126, 52, 127, 53, 0, 54, 1, 55, 2,
113    56, 3, 57, 4, 43, 5, 45, 6, 42, 7,
114    47, 8, 40, 9, 41, 10, 36, 11, 61, 12,
115    32, 13, 44, 14, 46, 15, 35, 16, 91, 17,
116    93, 18, 37, 19, 34, 20, 95, 21, 33, 22,
117    38, 23, 39, 24, 63, 25, 60, 26, 62, 27,
118    64, 28, 92, 29, 94, 30, 59, 31/
119    K=0
120    FOR(J=1; J<=MAX1 & K<=MAX0 ; INCREMENT(J) ) [
121    INCREMENT(K)
122    IF(STR1(J) == DHAT ) [
123    INCREMENT(J)
124    STR0(K)=DSPLY(2,STR1(J)+1)
125    ]
126    ELSE
127    IF(STR1(J) == DAT) [
128    INCREMENT(J)
129    STR0(K)=DSPLY1(STR1(J)+1)
130    ]
131    ELSE
132    STR0(K) = DSPLY(1,STR1(J)+1) ]
133    INCREMENT(K)
134    STR0(K)=EOS
135    RETURN

```

136 END

```

#
# OUTMAP - CONVERT ASCII CHAR SET TO DISPLAY CODE
#HJ DUNN, APRIL 14, 1981
#
SUBROUTINE OUTMAP(STRI,I,STRO,O)
  INTEGER STRI(DUMMYSIZE),I, # INPUT STRING ( LEN=I )
  STRO(DUMMYSIZE),O, # OUTPUT STRING ( LEN=O )
  DISPLY(128), # DISPLAY CODE OUTPUT CHAR
  J,C
  DATA DISPLY /1R5,1R6,1R7,1R8,1R9,1R+,1R-,1R*,1R/,1R(,
1R),1R3,1R.,1R.,1R.,1R.,1R[,1R],1R%,
1R",1R_,1R!,1R$,1R',1R?,1R<,1R>,74B,1R\,
75B,1R:,1R;,1R!,1R",
1R#,1R$,1R%,1R$,1R',1R(,1R),1R*,1R+,1R.,
1R-,1R.,1R/,1R0,1R1,1R2,1R3,1R4,1R5,1R6,
1R7,1R8,1R9,1R:,1R;,1R<,1R>,1R?,74B,
1RA,1RB,1RC,1RD,1RE,1RF,1RG,1RH,1RI,1RJ,
1RK,1RL,1RM,1RN,1RO,1RP,1RQ,1RS,1RT,
1RU,1RV,1RW,1RX,1RY,1RZ,1R[,1R],1R\,76B,
1R_,1R:,1RA,1RB,1RC,1RD,1RE,1RF,1RG,1RH,
1RI,1RJ,1RK,1RL,1RM,1RN,1RO,1RP,1RQ,1RR,
1RS,1RT,1RU,1RV,1RW,1RX,1RY,1RZ,1R0,1R1,
1R2,1R3,1R4/
  O=0
  DO J=1,I [
    C=STRT(J)
    IF(C >= CNTLA & C <= DEL) [
      IF(C < BLANK ++ C > UNDERLINE ) [
        INCREMENT(O)
        STRO(O)=DHAT
      ]
      INCREMENT(C)
      INCREMENT(O)
      STRO(O)=DISPLY(C)
    ]
  ]
  RETURN
END

```

SYMBOLIC CONSTANT = DEFINITION

1	3AD = -1
2	BLANK = 32
3	BUFFERLEN = 246/82
4	CNTLA = 0
5	DAT = 74B
6	DECREMENT = \$=\$-1
7	DFL = 127
8	DHAT = 763
9	DUMMYSIZE = 1
10	EOF = -3
11	EWS = C
12	FORMFEED = 12
13	HUGE = 32767
14	INCREMENT = \$=\$+1
15	MAX = 80
16	MAXDISPLAY = 81
17	MAXLINE = 91
18	STOIN = 5
19	STOOUT = 6
20	TAPEND = TAPE\$
21	UNDERLINE = 95
22	YES = 1

PROGRAM FOR RATFOR EXAMPLE /FO/SY

```

      C PROGRAM FOR RATFOR EXAMPLE /FO/SY
1      PROGRAM EXRAT ( INPUT = 246 / 82, OUTPUT = 246 / 82, TAPE 5 =
      $INPUT, TAPE 6 = OUTPUT )
2      INTEGER IO, BUFFER ( 81 )
3      INTEGER STRPUT, STRGET
4      INTEGER START (15)
5      DATA START /83, 84, 65, 82, 84, 32, 79, 70, 32, 73, 78, 80, 85,
      $84, 0 /
6      IO = STRPUT ( 6, START )
7      20000 CONTINUE
8      IO = STRGET ( 5, BUFFER, 80 )
9      IF (.NOT.( IO .EQ. - 3 )) GOTO 20003
10     GOTO 20002
11     20003 CONTINUE
12     IO = STRPUT ( 6, BUFFER )
13     20001 GOTO 20000
14     20002 CONTINUE
15     WRITE ( 6, 1 )
16     1      FORMAT ( 10X, 12HEND OF INPUT )
17     STOP
18     END

      C
      C SLEN - COMPUTE LENGTH OF STRING
      C
19     INTEGER FUNCTION SLEN ( STR )
20     INTEGER STR ( 1 )
21     DO 20005 SLEN = 1, 32767
22     IF (.NOT.( STR ( SLEN ) .EQ. 0 )) GOTO 20007
23     GOTO 20006
24     20007 CONTINUE
25     20005 CONTINUE
26     20006 CONTINUE
27     SLEN = SLEN - 1
28     RETURN
29     END

      C
      C STRPUT - WRITE A STRING TO A SPECIFIED LUN
      CHJ DUNN APRIL 14, 1981
      C
30     INTEGER FUNCTION STRPUT ( LUN, STR )

```



```

31      INTEGER LUN, I, N, SLEN, MINO, CEOF
32      INTEGER STR ( 1 ), BUFF ( 81 )
33      C
34      I = MINO ( 91, SLEN ( STR ) )
35      IF (.NOT.( STR ( 1 ) .EQ. 12 )) GOTO 20009
36      I = I - 1
37      IF (.NOT.( I .EQ. 0 )) GOTO 20011
38      WRITE ( LUN, 2 )
39      RETURN
40      20011 CONTINUE
41      CALL OUTMAP ( STR ( 2 ), I, BUFF, N )
42      WRITE ( LUN, 2 ) ( BUFF ( I ), I = 1, N )
43      FORMAT ( 141, 91 R1 )
44      GOTO 20010
45      20009 CONTINUE
46      IF (.NOT.( I .EQ. 0 )) GOTO 20013
47      WRITE ( LUN, 1 )
48      RETURN
49      20013 CONTINUE
50      CALL OUTMAP ( STR, I, BUFF, N )
51      WRITE ( LUN, 1 ) ( BUFF ( I ), I = 1, N )
52      FORMAT ( 14, 81 R1 )
53      20014 CONTINUE
54      20010 CONTINUE
55      IF (.NOT.( CEOF ( LUN ) .NE. 0 .OR. IOCHEC ( LUN ) .NE. 0 ))
56      $GOTO 20015
57      STRPUT = - 1
58      GOTO 20016
59      20015 CONTINUE
60      STRPUT = 1
61      20016 CONTINUE
62      RETURN
63      END
64      C
65      C STRGET - READ A STRING FROM A SPECIFIED LUN
66      C
67      C
68      INTEGER FUNCTION STRGET ( LUN, STR, MAXS )
69      INTEGER LUN, STR ( MAXS ), BUFF ( 81 )
70      READ ( LUN, 1 ) BUFF
71      FORMAT ( 81 R1 )

```

```

66      IF (.NOT.( CEOF ( LUN ) .NE. 0 )) GOTO 20017
67      STRGET = - 3
68      STR ( 1 ) = 0
69      RETURN
70 20017 CONTINUE
71      IF (.NOT.( IJCHEC ( LUN ) .NE. 0 )) GOTO 20019
72      STRGET = - 1
73      STR ( 1 ) = 0
74      RETURN
75 20019 CONTINUE
76      CALL INMAP ( BUFF, 81, STR, MAXS )
77      RETURN
78      END

C
C CEOF - HIDDEN EOF FUNCTION
CHJ DUNN, APRIL 14, 1981
C
79      INTEGER FUNCTION CEOF ( LUN )
80      INTEGER EOF,LUN
81      CEOF=EOF(LUN)
82      RETURN
83      END

C
C INMAP - CONVERT DISPLAY CODE TO INTERNAL ASCII
CHJ DUNN, APRIL 10, 1981
C
84      SUBROUTINE INMAP ( STRI, MAXI, STRO, MAXO )
85      INTEGER STRI ( 1 ), MAXI, STRO ( 1 ), MAXO, DSPLY ( 2, 64 ),
      $DSPLY1 ( 8 )
86      INTEGER J
87      DATA DSPLY1 / 0, 64, 94, 37, 58, 3 * 0 /
88      DATA DSPLY / 58, 96, 65, 97, 66, 98, 67, 99, 68, 100, 69, 101, 70,
      $, 102, 71, 103, 72, 104, 73, 105, 74, 106, 75, 107, 76, 108, 77,
      $109, 78, 110, 79, 111, 80, 112, 81, 113, 82, 114, 83, 115, 84,
      $116, 85, 117, 86, 118, 87, 119, 88, 120, 89, 121, 90, 122, 48,
      $123, 49, 124, 50, 125, 51, 126, 52, 127, 53, 0, 54, 1, 55, 2, 56,
      $ 3, 57, 4, 43, 5, 45, 6, 42, 7, 47, 8, 40, 9, 41, 10, 36, 11, 61,
      $ 12, 32, 13, 44, 14, 46, 15, 35, 16, 91, 17, 93, 18, 37, 19, 34,
      $20, 95, 21, 33, 22, 38, 23, 39, 24, 63, 25, 60, 26, 62, 27, 64,
      $28, 92, 29, 94, 30, 59, 31 /

```

C
C OUTMAP - CONVERT ASCII CHAR SET TO DISPLAY CODE
CHJ DUNN, APRIL 14, 1981

```
$      1R_,1R1,1RA,1RB,1RC,1RD,1RE,1RF,1RG,1PH,
$      1RI,1RJ,1RK,1RL,1RM,1RN,1RO,1PP,1RQ,1RR,
$      1RS,1RT,1RU,1RV,1RW,1RX,1RY,1RZ,1RO,1R1,
$      1R2,1R3,1R4/
```

```

      N = 0
      DO 20028 J = 1, I
      C = STRI ( J )
      IF (.NOT. ( C .GE. 0 .AND. C .LE. 127 )) GOTO 20030
      IF (.NOT. ( C .LT. 32 .OR. C .GT. 95 )) GOTO 20032
      D = D + 1
      STRO ( D ) = 76B
20032 CONTINUE
      C = C + 1
      D = D + 1
      STRO ( D ) = DISPLY ( C )
20030 CONTINUE
20028 CONTINUE
20029 CONTINUE
      RETURN
      END

```

REFERENCE

1. Kernighan, Brian W.; and Plauger, P. J.: Software Tools. Addison-Wesley Publishing Company, c. 1976.

1. Report No. NASA TM-83173		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle NASA LaRC RATFOR DOCUMENTATION VERSION 1.0				5. Report Date September 1981	
				6. Performing Organization Code 505-34-33-05	
7. Author(s) H. J. Dunn				8. Performing Organization Report No.	
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665				10. Work Unit No.	
				11. Contract or Grant No.	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546				13. Type of Report and Period Covered Technical Memorandum	
				14. Sponsoring Agency Code	
15. Supplementary Notes					
16. Abstract NASA LaRC implementation of the preprocessor RATFOR is described. RATFOR is a preprocessor that can be used to generate a well-structured program with source code that is easy to follow.					
17. Key Words (Suggested by Author(s)) RATFOR structured programing preprocessor FORTRAN			18. Distribution Statement Unclassified - Unlimited Subject Category 61		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 20	22. Price* A02		

End of Document